Notes

Lab06 due tonight

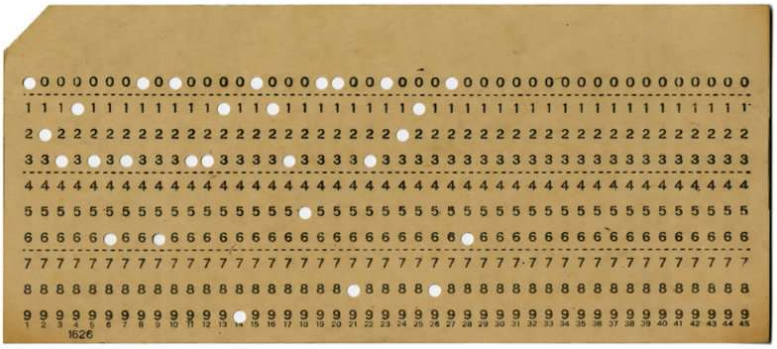
Quiz Weds

Still do not know date for this course's final. I am pushing Dean's office.

Midterm Oct 5

- My Oct 5 office hours will move to Oct 4, 2:45-3:45, via Zoom

User Input Handling



How do users enter input into a computer?
- keyboard, mouse, touchscreen, data network

Picture shows a computer punch card. People would type in characters on a special machine that would punch holes in the cards. Computers had "punch card readers" that would read big stacks of the cards, which would hold your programs and data. Even your compilers.

These predate me by about 20 years.

Bit of computer history
- First "mechanical adding machine in 1600s. Other interesting mechanical computers included:
    - Programmable looms for weaving patterns into fabric
    - Analog computers for doing integration or solving differential equations
    - Artillery sighting
- First "all-electronic digital computer" maybe ENIAC, 1945
- First PCs about 1980

User Input

`InputStream System.in` – delivers input from keyboard as a "stream"

`class Scanner(InputStream)` - scans the input stream and returns different types of data

A few methods:

- `String Scanner.next()` – returns the next "word" (up to a space)
- `boolean Scanner.nextBoolean()`
- `byte Scanner.nextByte()`
- `double Scanner.nextDouble()`
- `String Scanner.nextLine()` – returns text up to the next End-Of-Line

Talked about input from cmd line. Now from keyboard
We won't dive into streams now. It's an object we can keep reading from.
Need import java.util.Scanner;
IMPORT like in Python. Include modules so you can use them in your software

Reading User Input

The three key lessons for the semester:

- **Design your classes thoughtfully so they are useful and reusable**
- **Spend as much time thinking about testing as you do about your software design; spend as much time testing as you spend coding**
- **Never trust user input**

Reading User Input

A few more useful `Scanner` methods:

- `boolean Scanner.hasNext()` – is there an input word available?
- `boolean Scanner.hasNextInt()` – input int available?
- `boolean Scanner.hasNextFloat()` – input float available?
- `boolean Scanner.hasNextLine()` – is there a line available?

See ReadKeyboard.java
Did the user give you what you asked for?

Scanner

Scanner can read from a `String` or a `File` also

More Robustness

What if we don't have enough input?

Summary:

- **Must validate that TYPE of input is what is expected**
- **Must validate that we actually have input**

Input Redirection

Instead of actual typed input data, the Operating System let's us pretend to send keyboard input to a program while actually taking it from a file.
So when I'm testing 60 students' homework this week, I'm not actually going to type in the same damn thing 60 times.

Redirection an "Operating System thing", not Java specific

```
% cat myFile
11 22 33 44 55 66

% java ReadKeyboard < myFile
Read an int:  11
Read an int:  22
Read an int:  33
Read an int:  44
Read an int:  55
Read an int:  66
```

It's the "<" character
NO COMMAND LINE ARGUMENTS! The OS handles this before calling the program.

Can save output to a file

```
% java ReadCommandLine 11 22 33 44 55 66 > outputFile

% cat outputFile
Read an int:  11
Read an int:  22
Read an int:  33
Read an int:  44
Read an int:  55
Read an int:  66
```

It's the ">" character
Can do both < and >, of course.

System.err

A second object that prints to the screen, for error reporting

```
System.err.println("Your input is bad…and so are you!");
```

When we redirect output, can direct System.out ("stdout") and System.err ("stderr") separately.

GO TRY IT in demo code!

CS112 –
Java
Programming

Fall 2022

Loops and Conditionals

Loops (and Conditionals) in Java

What have we been introduce to so far?

```
while(boolean) { }


if(boolean) {
} else if (boolean) {
}
```

<<more else-if statements, optionally>>

```
} else {
}
```

The switch Statement

The *switch statement* provides another way to decide which statement to execute next

The `switch` statement evaluates an expression, then attempts to match the result to one of several possible *cases*

Each case contains a value and a list of statements

The flow of control transfers to statement associated with the first case value that matches

The switch Statement

The general syntax of a `switch` statement is:

`switch` and `case` are reserved words

```
switch ( expression )
{
   case value1 :
      statement-list1
   case value2 :
      statement-list2
   case value3 :
      statement-list3
   case  ...

}
```

If *`expression`* matches *`value2`*, control jumps to here

The switch Statement

Often a *break statement* is used as the last statement in each case's statement list

A `break` statement causes control to transfer to the end of the `switch` statement

If a `break` statement is not used, the flow of control will continue into the next case

Sometimes this may be appropriate, but ALMOST ALWAYS we want to execute only the statements associated with one case. So we use the `break` statement

break; can break out of a while loop also! Or ANY type of loop

The switch Statement

An example of a switch statement:

```
switch (option)
{
   case 'A':
      aCount++;
      break;
   case 'B':
      bCount++;
      break;
   case 'C':
      cCount++;
      break;
}
```

The switch Statement

A `switch` statement can have an optional *default case*

The default case has no associated value and simply uses the reserved word `default`

If the default case is present, control will transfer to it if no other case value matches

If there is no default case, and no other value matches, control falls through to the statement after the switch

The switch Statement

The type of a switch expression must be integers, characters, or enumerated types

As of Java 7, a switch can also be used with strings

You cannot use a switch with floating point values

The implicit boolean condition in a `switch` statement is equality

You cannot perform relational checks with a `switch` statement

See `GradeReport.java`

```java
//********************************************************************
//  GradeReport.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a switch statement.
//********************************************************************

import java.util.Scanner;

public class GradeReport
{
   //-----------------------------------------------------------------
   //  Reads a grade from the user and prints comments accordingly.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      int grade, category;

      Scanner scan = new Scanner(System.in);

      System.out.print("Enter a numeric grade (0 to 100): ");
      grade = scan.nextInt();

      category = grade / 10;

      System.out.print("That grade is ");
```

continue

continue

```
      switch (category)
      {
         case 10:
            System.out.println("a perfect score. Well done.");
            break;
         case 9:
            System.out.println("well above average. Excellent.");
            break;
         case 8:
            System.out.println("above average. Nice job.");
            break;
         case 7:
            System.out.println("average.");
            break;
         case 6:
            System.out.println("below average. You should see the");
            System.out.println("instructor to clarify the material "
                               + "presented in class.");
            break;
         default:
            System.out.println("not passing.");
      }
   }
}
```

continue

Sample Run

```
Enter a numeric grade (0 to 100): 91
That grade is well above average. Excellent.
```

```
      swi
      {
            System.out.println ("a perfect score. Well done.");
            break;
         case 9:
            System.out.println ("well above average. Excellent.");
            break;
         case 8:
            System.out.println ("above average. Nice job.");
            break;
         case 7:
            System.out.println ("average.");
            break;
         case 6:
            System.out.println ("below average. You should see the");
            System.out.println ("instructor to clarify the material "
                                + "presented in class.");
            break;
         default:
            System.out.println ("not passing.");
      }
   }
}
```

The Conditional Operator

The *conditional operator* evaluates to one of two expressions based on a boolean condition

Its syntax is:

```
condition ? expression1 : expression2
```

If the ***condition*** is true, ***expression1*** is evaluated; if it is false, ***expression2*** is evaluated

The value of the entire conditional operator is the value of the selected expression

The Conditional Operator

The conditional operator is similar to an `if-else` statement, except that it is an expression that returns a value

For example:

```
larger = ((num1 > num2) ? num1 : num2);
```

If `num1` is greater than `num2`, then `num1` is assigned to `larger`; otherwise, `num2` is assigned to `larger`

The conditional operator is *ternary* because it requires three operands

The Conditional Operator

Another example:

```
System.out.println("Your change is " + count +
   ((count == 1) ? "Dime" : "Dimes"));
```

If `count` equals 1, the "Dime" is printed

If `count` is anything other than 1, then "Dimes" is printed

The do Statement

A *do statement* has the following syntax:

```
do
{
    statement-list;
}
while (condition);
```

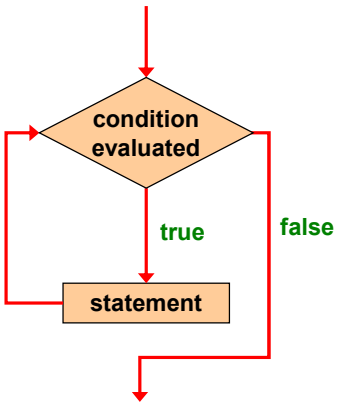
The **`statement-list`** is executed once initially, and then the **`condition`** is evaluated

The statement is executed repeatedly until the condition becomes false

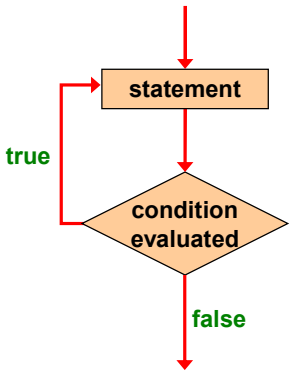
I HAVE NEVER USED A DO-WHILE STATEMENT IN MY LIFE. I JUST USE WHILE

Comparing while and do

The while Loop



The do Loop



The for Statement

A *for statement* has the following syntax:

The *initialization* is executed once before the loop begins

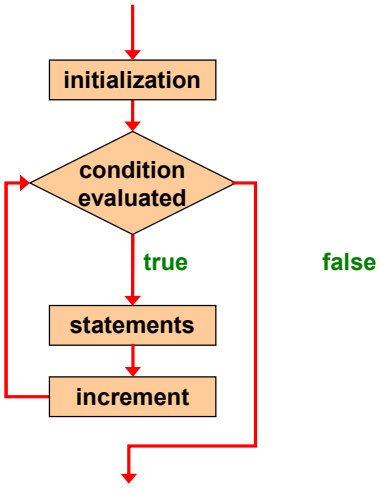
The *statement* is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment ) {
   statements;
}
```

The *increment* portion is executed at the end of each iteration

Different from Python!

Logic of a for loop



The for Statement

A `for` loop is functionally equivalent to the following `while` loop structure:

```
initialization;
while ( condition )
{
   statement;
   increment;
}
```

The for Statement

An example of a `for` loop:

```
for (int count=1; count <= 5; count++)
   System.out.println(count);
```

The initialization section can be used to declare a variable

Like a `while` loop, the condition of a `for` loop is tested prior to executing the loop body

Therefore, the body of a `for` loop will execute zero or more times

The for Statement

The increment section can perform any calculation:

```
for (int num=100; num > 0; num -= 5)
   System.out.println(num);
```

A `for` loop is well suited for executing statements a specific number of times that can be calculated or determined in advance

See `Stars.java`

```java
//********************************************************************
//  Stars.java       Author: Lewis/Loftus
//
//  Demonstrates the use of nested for loops.
//********************************************************************

public class Stars
{
   //-----------------------------------------------------------------
   //  Prints a triangle shape using asterisk (star) characters.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      final int MAX_ROWS = 10;

      for (int row = 1; row <= MAX_ROWS; row++)
      {
         for (int star = 1; star <= row; star++)
            System.out.print("*");

         System.out.println();
      }
   }
}
```

```
//************************************************************************
//  Stars.java       Auth                                              s
//
//  Demonstrates the use                                              oops.
//************************************************************************

public class Stars
{
   //-----------------------------------------------------------------------
   //  Prints a triangle                                                erisk (star) characters.
   //-----------------------------------------------------------------------
   public static void mai                                              s)
   {
      final int MAX_ROWS

      for (int row = 1; row <= MAX_ROWS; row++)
      {
         for (int star = 1; star <= row; star++)
            System.out.print("*");

         System.out.println();
      }
   }
}
```

Output

```
*
**
***
****
*****
******
*******
********
*********
**********
```

The for Statement

Each expression in the header of a `for` loop is optional

If the initialization is left out, no initialization is performed

If the condition is left out, it is always considered to be true, and therefore creates an infinite loop

If the increment is left out, no increment operation is performed

Nested Loops

```
for (int i= 0; i < height; i++) {
 for(int j = 0; j < width; j++) {
      totalBrightness +=image.pixelValue(i, j);
 }
}
```

Start with 0, end with one-less-than size. Because that is how array indexing works!

Boolean "early termination"

```
double numer = -1.0, denom = 0.0;

if (denom != 0.0 && (numer/denom) > 1.0) {
      System.out.println("Value is greater than 1");
}

if (numer < 0.0 || (numer/denom) < 0.0) {
      System.out.println("Error");
}
```

Since denom == 0.0, Java never evaluates (numer/denom) for the “&&” case.

How about this?

```
int y = 10, z = 5;
for (int x = 0; y > 0; z = x) {
      x++;
      y = z - x;
}
```

WALK THROUGH
AWFUL CODE: NEVER DO THIS. Try to have only a single variable inside for loop, and change and check only it

Summary

Conditionals

- if else-if else
- "conditional"

Loops

- while
- do-while
- for
- switch-case-break

Lab09

Talk about Fractions.java